Journal of Universal Computer Science, vol. 14, no. 19 (2008), 3160-3173
submitted: 29/10/08, accepted: 29/10/08, appeared: 1/11/08

Model-Driven Approach to Design User Interfaces for Workflow Information Systems

Josefina Guerrero García
(Université catholique de Louvain, Louvain-la-Neuve, Belgium
josefina.guerrero@student.uclouvain.be)

Christophe Lemaigre
(Université catholique de Louvain, Louvain-la-Neuve, Belgium
christophe.lemaigre@uclouvain.be)

Juan Manuel González Calleros
(Université catholique de Louvain, Louvain-la-Neuve, Belgium
juan.gonzalez@student.uclouvain.be)

Jean Vanderdonckt
(Université catholique de Louvain, Louvain-la-Neuve, Belgium
jean.vanderdonckt@uclouvain.be)

Abstract: Many methods in the area of Human-Computer Interaction have been developed for deriving user interfaces considering individual users. However, nowadays information systems include more than single interaction, there is a need to explicitly include multi user interaction during the design process of information systems. One of this approaches are workflow systems. Workflow information systems present the view of the organization, while modelling their business process; workflow systems define explicitly the role of each actor during the performance of the different tasks. The introduction of aspects such as: organizational units, agendas, Wok list, user stereotypes and resources, allows the design of more robust systems, especially when all of them are considered during the development of the User Interfaces needed. In this paper, we present a model-driven approach to derive user interfaces of a workflow information system from a series of models. A graphical editor has been developed. It is described and exemplified on a real-world case study for designing the user interfaces of a workflow information system.

Keywords: Model-driven approach, workflow information systems, user interface development, collaboration, user interfaces flow.
Categories: H.1.0, H.4.0, H.5.2, H.5.3

1 Introduction

Workflow applications are used in organizations to increase efficiency and compliance of important business processes. In the same venue, Workflow Information Systems (WIS) refer to the application of information technology to business problems. Its primary characteristic is the automation of processes involving combinations of human activities with information technology applications. Owing to the fact that the users of an information system interact with it through its user

interfaces (UIs) in the pursuit of organizational goals, users require flexibility when interacting with the system to contend with changes in the business processes, to support differing work approaches, and to coordinate the activities of various users, flexibility in creating UIs is therefore important [Stavness, 04]. In the area of Workflows, there has been increasing interest in developing methods and languages to design UIs, as the work of [Kristiansen, 07] and [Brandl, 01]. In Human-Computer Interaction, the user interfaces description languages (UIDLs) express various aspects of the UI, including the abstract and concrete elements of the UI, the tasks to be performed by the user, and the UI dialogue. Nevertheless, there are several opportunities to improve on these UIDLs; in this paper, we will explore a systematic way to design UIs for a WIS. The workflow model defines what processes and tasks need to be fulfilled and their possible ordering; hence the workflow model is a 'framework' for creating task model, hence the task model is suitable for designing UIs. This model-driven approach tries to provide some guidelines to design UIs for WIS.

The remainder of this paper is structured as follows: Section 2 presents an overview of the related work. Section 3 introduces the workflow model proposed. Section 4 illustrates the way in which UIs of a workflow are designed. Section 5 introduces a case study using a tool support. Section 6 summarizes our work, deriving conclusions and addressing future work.

2 Related Work

In this section we will give a short introduction to workflow models, task modelling, and describe how task models and XML-based languages have been used to develop UIs.

2.1 Workflow Models

Workflow models focus on how work is done to accomplish organizational goals; it defines how task, information, and documents are passed from one participant to another in the organization [WfMC, 06]. The introduction of WIS in organizations has emerged as a major advantage to provide them with a competitive advantage, as a way to trim cost, to automate process, to reduce time. Implemented properly, workflow applications enable companies to reengineer and streamline business processes; for this reason, the interest in workflow systems has grown dramatically over the last years. We distinguish the term *business process,* widely used in several researches, from WIS. In fact *business process* modeling do not necessarily is devoted or limited to the automation of business process using information systems. Workflow models include different notations, languages, and software tools. As notations there are Petri Nets [van der Aalst, 98], Statecharts Diagrams [W3C, 05] [Wodtke, 97], Business Process Modeling Notation (BPMN) [OMG, 06], and UML Activity Diagrams [Dumas, 01]. Currently, several models and design methods support the development of complex workflow-based applications providing notations for business process and including tool support. They are: The Progression Model [Stavness, 04], YAWL [van der Aalst, 05], Microsoft Windows Workflow Foundation (WWF) [Esposito, 05], WebSphere® MQ Workflow (IBM) [IBM, 06],

and WIDE [Casati, 96]. Due to the large amount of existing workflow products we came to a point where it is very difficult to analyze and compare their capabilities on a common scheme. However, they can be gathered in a collection of workflow patterns that provide the basis for an in-depth comparison of commercially available workflow systems. Control-flow patterns [van der Aalst, 03] identified useful basic routing constructs such as sequence, parallel split, synchronization, exclusive choice. From a data perspective, there is a series of characteristics that occur repeatedly in different workflow modeling paradigms. Workflow data patterns [Russell, 04] are aimed at capturing the various ways in which data is represented and used in workflows. Workflow resource patterns [Russell, 05] correspond to the manner in which tasks are allocated to resources, that is any entity that is capable of achieving some work unit.

2.2 Task Models

Task models are logical descriptions of activities that are designed to be carried out in reaching user's goals in an interactive system. There are many different languages to task modeling, such as: Hierarchical Task Analysis (HTA) [Annett, 04] describing the goals, tasks and operations in logical structures of different levels. Task Knowledge Structure (TKS) [Johnson, 91] which is a conceptual representation of the knowledge a person has stored in her memory about a particular task. GroupWare Task Analysis (GTA) [van de Veer, 99] was developed as a means to model he complexity of tasks in a cooperative environment. ConcurTaskTrees (CTT) [Paternò, 99] notation was created to support engineering approaches to task modeling. All of them support designers by hierarchically decomposing tasks, defining objects manipulated and the role responsible for performing the task. The vast number of task modeling notations results in semantic and syntactic differences which are discussed in [Limbourg, 04a].

Task models play an important role in UI design because they support the systematic representation of the user activity as opposed to the system activity.

2.3 User Interface Description Languages

Model-based user interface design is intended to assist in designing UIs with a more formal computer supported methodology; a user interface description language (UIDL) is intended to capture the details of what a user interface could or should consist. There are solutions for developing UIs that are based in eXtensible Mark-up Language (XML) such as: User Interface Markup Language (UIML) [Abrams, 99] [Helms, 08], Abstract User Interface Markup Language (AUIML) [Azevedo, 00], eXtensible Interface Markup Language (XIML)[Eisenstein, 00], and Teresa XML [Paternò, 02] among others.

UsiXML [Limbourg, 04b] is a XML-compliant markup language capturing the essence of what a UI is or should be independently of physical characteristics. It has been selected as the UIDL to be used in the remainder of this work because of its capabilities of extensiveness, availability, central storage of models, and its model-driven approach.

3 A Workflow Metamodel

After reviewing the literature, we propose a metamodel that considers the principal components to model WIS. The intention is to use this model as a base to design the necessary UIs. Figure 1 reproduces a simplified version of the metamodel used in this work, a complete version of it can be found in [Guerrero, 08].

- Workflow model: It describes how the work in organization flows by defining models of: process (what to do?), tasks (how to do it?), and the organizational structure (where and who will perform it?). A workflow model has at least one process and each process has at least two tasks. The heuristics to identify a workflow model are: it is associated to the operational and/or administrative objectives of organization, it is performed within the same organization and it is associated to the automation of a business process. Workflows are described with a name and ID.
- Process Model: The definition of a process indicates the ordering of tasks in time, space, and resources. Our model is an adaptation of the Petri net notation proposed in [van de Aalst, 02] and it is compatible with the workflow resource patterns proposed in [Russell, 05]. The process model is composed of a target, a source, operators, and work item. The concept of *work List* is introduced, which stocks the processes of the whole organization. Managers are benefited as they can identify resources performing tasks, status of the workflow, bottlenecks in the processes and the identification of the organizational unit where the task is performed. The heuristics to identify a process model are: same group of resources, continuous period of time, specific ordering of tasks, the work is developed within groups, among groups or by a group as a whole, is not further divided into sub-processes and it could be primary (production), secondary (support), or tertiary (managerial).
- Task model: An adapted version of CTT [Paternò, 99] is used in this work. A task is an activity that has to be performed by users (human, systems, humans interacting with systems or a combination of them) to reach a given goal related to the business processes. Introducing task models description to the workflow models corresponds, but is not limited, to the following reasons:
  1. Task models describe, opposed to process models, end users' view of interactive tasks while interacting with the system. This allows describing how a task is performed.
  2. It is true that in a process model we can add the detail desired, with process hierarchies, to represent a detailed task description. However, we consider that specific temporal operators, iteration, suspend/ resume, applied to task, can be more naturally defined in a task model rather into a process model, that implies the creation of dummy transitions.

  The heuristic to identify a task are: same place, same type of resource, same period of time, and the work is developed by one resource (individual), it could be user, interactive, system or abstract task. Based on the organizational model, we can add a machine task (develop by any mechanical or electrical device that transmits or modifies energy to perform or assist in the performance of tasks. For instance: fax, robot.). The task model is composed of target, source, task, relationships, decomposition, and temporal relationships.

- Organizational model. It describes the places were work is performed, the users that perform the work, and so on. This part contributes to UI adaptation to different categories of users and security of IS by blocking access to UIs when the user does not have the permission to perform the task. An *organizational Unit* describes a formal group of people working together with one or more shared goals or objectives. It could be composed of other organizational units. Inside these units *a task resource* is directly or indirectly involved in carrying out the work. The *LogEntry* describes specific characteristics of the resources. Each resource may have a log Entry associated with them. A *Job* represents the total collection of tasks, duties, and responsibilities assigned to one or more positions which require work of the same nature and level, for instance, a surgeon. At this level an *Agenda* is defined showing assigned tasks to the user. It allows the description of the different status of a task (for instance: not started, in progress), the date when the task begins, the deadline, the date when the task could be assigned or delegated, and the date when the task is completed.

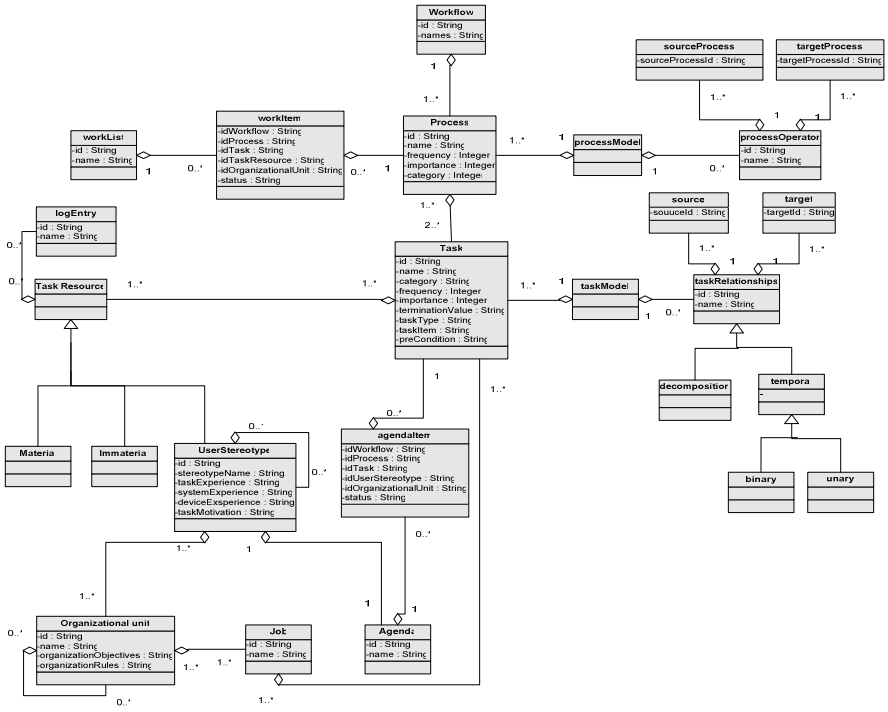

Figure 1: Simplified view of the metamodel

In [Guerrero, 08b] we have introduced a set of precise criteria that can be used in order to identify a task and to distinguish a task from other concepts like process and workflow, which are located at another level in the hierarchy, but at the same level of abstraction.

In a model-driven approach all the components are models. Even transformation among models and relationships are described in terms of a metamodel. The mapping model defines the relationships between the models. This mapping model allows the specification of the link of elements from heterogeneous models and viewpoints. Several relationships can be defined to explicit the relationships between models. We extended the existing mapping model of UsiXML describing task execution (rules to specify: complex and dynamic users' interaction within the organization), such as: Is Grafted On mapping, this relationships is useful when a task (Tj) has been executed, and a task complementary (Ti) is defined to realize the first task where Ti is completely autonomous to Tj. When work is executed tasks are defined by a userStereotype. Then, they can be allocated to task Resources, following the set of predefined workflow resource patterns, proposed in [van der Aalst, 05]. These patterns represent the different ways in which tasks are advertised and ultimately bound to specific resources for execution.

Recently there has been increasing interest in developing UIs for group tasks in multi-user situations. In order to represent group's requirements to coordinate their work among themselves by relying on implicit (e.g., manual, verbal, informal) communication schemes, it is necessary to addressing criteria for support group interactions [Mandviwalla & Olfman, 94], such as the following ones we selected in our work:

- "Support carrying out group tasks" from the individual level continuously throughout the global level: individual, within groups, for the group as a whole, among groups, within organization, and among organizations.
- "Support multiple ways to support a group task": in principle, there should not be unique way to carry out a single group task, but several mechanisms should be offered for this purpose. If a mechanism is no longer available, another one should be selectable.
- "Support the group evolution over time": when the group evolves over time, the workflow definition should be easily maintained and reflected in the system.

4 How to design the Workflow User Interfaces

Model-based user interface design processes often start with a task related model that is evolved through an incremental approach to the final UI [Cuppens, 06]. In this way, we selected UsiXML to generate UIs [Vanderdonckt, 05]; it relies on the Cameleon Reference Framework [Calvary, 03]. The simplified version, reproduced in Figure 2, structures four development steps: 1) *Task & Domain*: describe the various user's tasks to be carried out and the domain-oriented concepts as they are required by these tasks to be performed. 2) *Abstract UI* (AUI): defines abstract containers and individual components, two forms of Abstract Interaction Objects by grouping subtasks according to various criteria (e.g., task model structural patterns, cognitive load analysis, semantic relationships identification), a navigation scheme between the container and selects abstract individual component for each concept so that they are independent of any modality. An AUI abstracts a CUI into a UI definition that is independent of any modality of interaction. 3) *Concrete UI* (CUI): concretizes an abstract UI for a given context of use into Concrete Interaction Objects (CIOs) so as

to define widgets layout and interface navigation. It abstracts a FUI into a UI definition that is independent of any computing platform. 4) *Final UI* (FUI): is the operational UI, i.e. any UI running on a particular computing platform.
To support conceptual modeling of UIs and to describe UIs at various levels of abstractions, the following models have been involved: **taskModel**: described in section 3. **domainModel**: is a description of the classes of objects manipulated by a user while interacting with a system. **mappingModel**: is a model containing a series of related mappings between models or elements of models. **contextModel**: is a model describing the three aspects of a context of use in which a end user is carrying out an interactive task with a specific computing platform in a given surrounding environment. Consequently, a context model consists of a *user model*, a *platform model*, and an *environment model*. **auiModel**: is the model describing the UI at the abstract level as previously defined. **cuiModel**: is the model describing the UI at the concrete level as previously defined. **transformationModel**: Graph Transformation (GT) techniques were chosen to formalize explicit transformations between any pair of models (except from the FUI level), because it is *visual* (every element within a GT based language has a graphical syntax), *formal* (GT is based on a sound mathematical formalism and enables verifying formal properties on represented artefacts), *seamless* (it allows representing manipulated artefacts and rules within a single formalism). **uiModel**: is the topmost superclass containing common features shared by all component models of a UI.

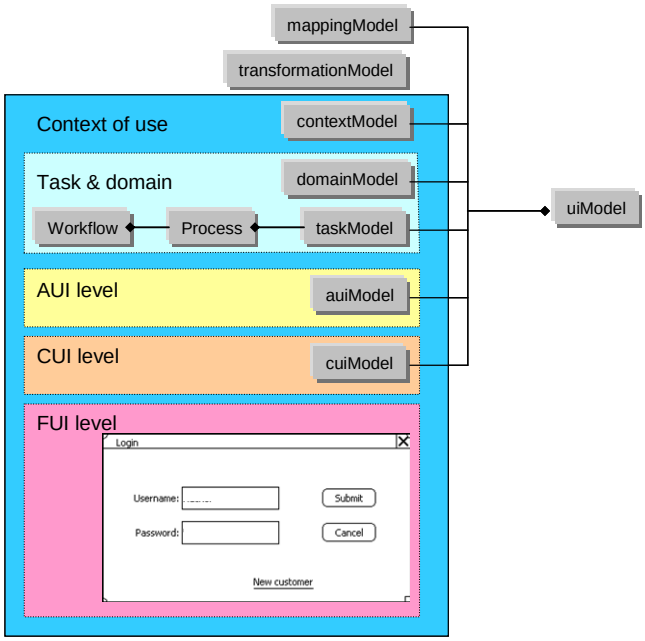

Figure 2: UsiXML and Cameleon Reference Framework

UsiXML describes at a high level of abstraction the constituting elements of the UI of an application: widgets, controls, containers, modalities, interaction techniques, etc. It supports device independence: a UI can be described in a way that remains autonomous with respect to the devices used in the interactions such as mouse, screen, keyboard, voice recognition system, etc. Language provides a means of communication; linguistics—the science of languages—can also be applied to programming languages. By this reason, we introduced the abstract syntax with "*enriched" directed graph*. That is to say an identified, labelled, typed, constrained graph. Finally we presented the stylistics in a graphical representation for the model presented in Figure 1 which relies on icons (Figure 3). Reusing this mechanics the UIs of a workflow model, that includes task model, can be generated.

As we said before, the aim of this work is to design the UIs of a workflow information system from its specifications (provided that they address the relevant aspects) and the different constructs that link the tasks and the users together.

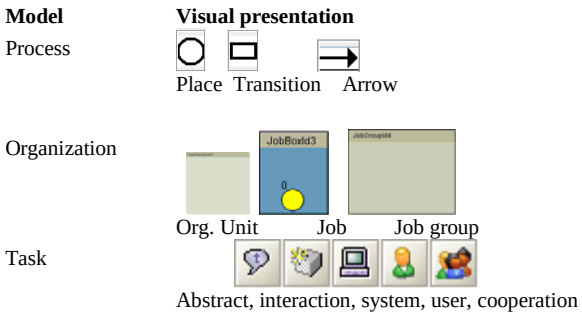

Figure 3: Stylistic representation

5 Case Study and Tool Support

In order to support the generation of UIs from a workflow model, a workflow editor has been developed. The case study will be elaborated and described among to the use of the tool and the concepts involved in each phase. During the stage of system requirements gathering, a model elicitation is used [Lemaigre, 08]. The model elicitation is aimed at identifying in textual scenarios elements that are relevant for building a first version of models that will be further exploited in a model-driven approach. Three method levels are successively examined to conduct model elicitation from textual scenarios for the purpose of conducting model-driven approach for designing user interfaces: manual classification, dictionary-based classification, and nearly natural language understanding based on semantic tagging and chunk extraction. The model elicitation process discussed in this case study involves the identification of several models: user, task, domain, organization, resources, and job. The case study analyzes how people take courses to get a driving license; see in Figure 4 how the model elicitation tool is used to analyze the textual definition of the case study. The name that represents an instance of a model element belonging to the ontology can be manually selected, highlighted, and assigned to the

corresponding concept, such as a task, a job, an organizational unit, etc. Consequently, all occurrences of this instance are automatically identified in the scenario and highlighted in the colour assigned to this concept. After this elicitation, tasks, jobs, organizational units, and resources are identified.

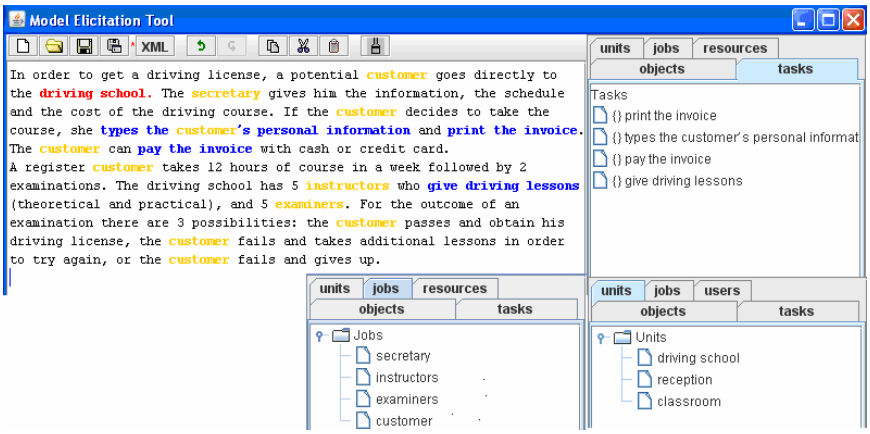

Figure 4: Elicitation task

In the *reception*, *Ann Scott* (*Secretary*) is in charge to receive all the *customers*, *give information* about the courses and payments. If the customer is interested, she *types the data* in the system and sends to *print the invoice* which the *customer pays*. After, the customer *takes the theorethical lessons* in a *classroom*, and *practical lessons* on the *street* with one of the *instructors*. In order to get the driving license, the customer needs to *aprove* a theoretical and a practical *test* which are applied by an *examiner*. If the customer pass both test, the *manager*, *Jeremy Burker*, *prepares the driving license* in his *office*. After one day, the customer can *pick up his driving license* on the reception.

Once each component of the process has been identified, then all the concepts are represented in a workflow model. The workflow editor, Figure 5, graphically represents four organizational units (reception, classroom, street and manager's office), where the different resources perform their tasks based on their jobs qualities (secretary, instructor, examiner, manager).

Defining resources and their jobs in the organization it is an important aspect that we consider. It is not just about identifying different jobs (manager, secretary, etc.) and resources involved in the process but also, attributes of the job (specification, family, grade, privileges) and resource (level of experience, hierarchy level) that can lead to further define who is capable to perform a determined task. For instance, we might consider that any examiner can apply the classroom assessment but just experienced can apply the street test. After defining tasks, jobs, and resources, tasks are allocated or offered to resources in different ways: direct allocation, delegation, history-based allocation, among others.

This assignation, Figure 6, is elaborated after the analysis of the characteristics (qualifications, skills, abilities, experience, and hierarchical level) of each resource and considering the requirements of each task.

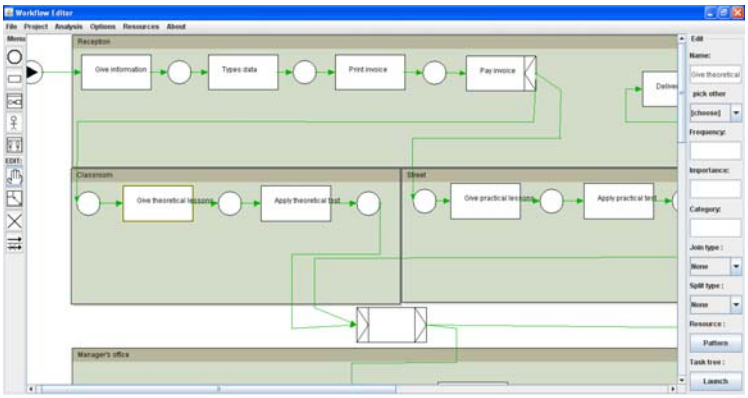

Figure 5: Workflow editor

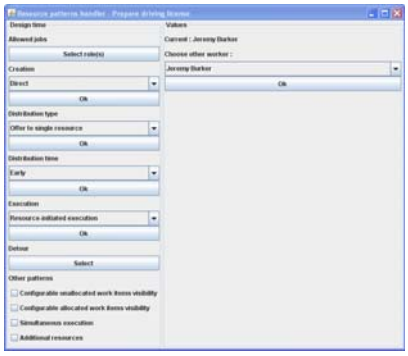

Figure 6: Assigning task to resource

For instance, if the manager is not available, the examiner who applied the test can elaborate the driving license. Once workflow components have been identified and specified on the workflow editor, we can detail in deep the steps to carry out tasks using task models. For instance, to execute the task "Type data", it is necessary to develop a series of sub-tasks in order to accomplish the goal, i.e. the secretary needs to type name, address, phone number and e-mail of the customer in the system (Figure 7). All this sub-tasks are interaction tasks. However, we can found several types of task inside the process, as we identify on Figure 3. Our task model tool is IdealXML [Montero, 06].

Finally, UIs for the case study are derived using the model-driven approach of UsiXML [Limbourg, 04b], a set of *transformation rules* were applied [see www.usixml.org for more information]. It is not the scope of this paper to address UI development but relies on existing work. UsiXML model-driven UI development uses

as input a task model, enriched with our meta-description, and transform it into a UI, in Figure 8 the type data task is depicted. Analogously, the remaining UIs are generated for each task model defined.

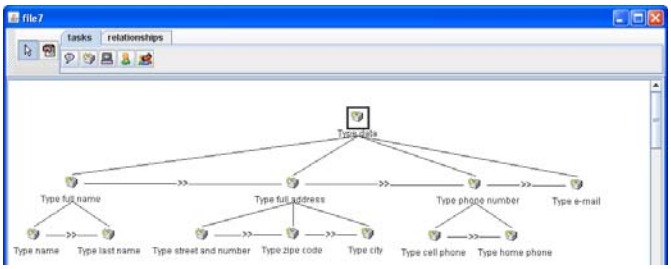

Figure 7: Task model

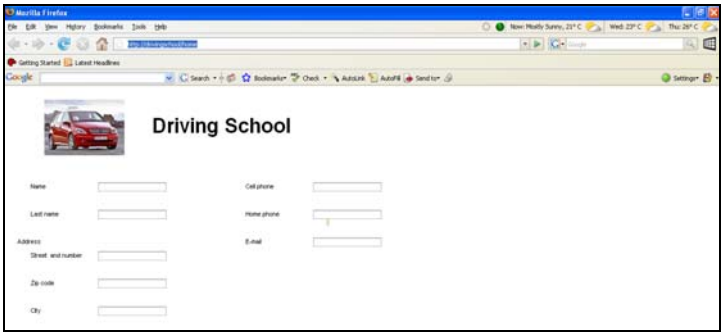

Figure 8: User interface derived from task model

6 Conclusion and Future Work

This paper presented a metamodel for designing the various user interfaces of a workflow information system, which are advocated to automate business processes, following a model-driven approach based on the requirements and processes of the organization. So far, the focus has been put mostly on the workflow information systems to support business process. For this purpose, a metamodel integrates the following concepts: workflow, process, task, domain, job definition, organizational structure, and resources.

A model-driven approach has been followed to progressively decompose the workflow model into processes which are in turn decomposed into tasks. From each task model, transformation rules were applied in order to design the different UIs involved in the workflow. A workflow tool has been developed to support the method enactment. The major benefit of the above method is that all the design knowledge required to progressively move from a workflow specification to its corresponding UIs is expressed in the metamodel and the mapping rules. The method preserves

continuity (all subsequent models are derived from previous ones) and traceability of its enactment (it is possible to trace how a particular workflow is decomposed into processes and tasks, with their corresponding user interfaces). In this way, it is possible to change any level (workflow, process, task, and UI) and to propagate the changes throughout the other levels by navigating through the mappings established at design time. The strengths of this work are: separation of concerns principle is respected; it bridges the gap between WIS and UI design, the steps of the model-driven approach define in a comprehensive way their logic and application. One identified difficulty is the need to go step by step. A case study has been reported and summarized to demonstrate the feasibility of this approach.

As future work, usability guidelines will be applied in the design of UIs, workflow analysis methods will be taken into account. Also, the Agenda where a resource is notified about the possible tasks that are allocated or offered to him will be designed.

FlowiXML web site

More information, including the tool, a video demo, link to other case studies, can be found at http://www.usixml.org/index.php?mod=pages&id=40. Information about UsiXML can be found at www.usixml.org. Information about IdealXML can be found at http://www.usixml.org/index.php?mod=pages&id=15.

Acknowledgements

We gratefully acknowledge the support of the CONACYT program (www.conacyt.mx) supported by the Mexican government and the SIMILAR network of excellence (http://www.similar.cc). Thanks to Francisco Montero for integrating IdealXML in this work. The authors also thank the anonymous reviewers for the comments on an earlier version of this manuscript.

References

[Abrams, 99] Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S., Shuster, J.: UIML: An Appliance-Independent XML User Interface Language. In A. Mendelson, Ed., Proc. of 8 International World-Wide Web Conference WWW's (Toronto, May 11-14, 1999), Amsterdam, 1999. Elsevier Science Publishers.

[Annett, 04] Annett, J.: Hierarchical Task Analysis. The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates, Mahwah, 2004. pp. 67-82.

[Azevedo, 00] Azevedo, P., Merrick, R., Roberts, D.:OVID to AUIML – user-oriented interface modeling. In N. Nunes, Ed., Proc. of 1 International Workshop "Towards a UML Profile for Interactive Systems Development" TUPIS'00 (York, October 23, 2000), York, 2000.

[Brandl, 01] Brandl, A.: Generating User Interfaces for Ad-hoc Workflows. Poster demo at IUI'01. Santa Fe, USA, 2001.

[Calvary, 03] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3 (2003), pp. 289–308

[Casati, 96] Casati, F. et al.: WIDE Workflow model and architecture. Technical Report 96-19, Centre for Telematics and Information Technology (CTIT), University of Twente.

[Cuppens, 06] Cuppens, E., Raymaekers, C., Coninx, K.: A model-based design process for interactive virtual environments. In Gilroy, S.W., Harrison, M.D. Eds. Interactive Systems. LNCS, vol. 3941, pp. 225-236. Springer, Heidelberg. 2006.

[Dumas, 01] Dumas, M. and ter Hofstede, A.: UML Activity Diagrams as a Workflow Specification Language. In M. Gogolla and C. Kobryn, editors, Fourth International Conference on the Unified Modeling Language (UML 2001), pp. 76-90, Toronto, Canada, 2001.

[Esposito, 05] Esposito, D.: Getting Started with Microsoft Windows Workflow Foundation: A Developer Walkthrough (2005). http://msdn.microsoft.com/winfx/reference/workflow

[Guerrero, 08a] Guerrero, J., Vanderdonckt, J., Gonzalez, J.M.: FlowiXML: a Step towards Designing Workflow Management Systems. Journal of Web Engineering 4, 2 (2008) 163–182.

[Guerrero, 08b] Guerrero, J., Vanderdonckt, J., Lemaigre, Ch.: Identification Criteria in Task Modeling, Proc. of 1st IFIP TC 13 Human-Computer Interaction Symposium HCIS'2008 (Milan, 8-9 September 2008), P. Forbrig, F. Paterno, A.M. Pejtersen (eds.), International Federation of Information Processing, Vol. 272, Springer, Boston, 2008, pp. 7-20.

[Helms, 08] Helms, J., Schaefer, R., Luyten, K., Vermeulen, J., Abrams, M., Coyette, A., Vanderdonckt, J.: Human-Centered Engineering with the User Interface Markup Language, in Seffah, A., Vanderdonckt, J., Desmarais, M. (Eds.), "Human-Centered Software Engineering", Chapter 7, HCI Series, Springer, London, 2008, pp. 141-173.

[IBM, 06] IBM WebSphere® MQ Workflow, 2006. http://www-306.ibm.com/software/integration/wmqwf/

[Johnson, 91] Johnson, P., and Johnson, H.: Task Knowledge Structures: Psychological basis and integration into system design. Acta Psychologica 78, pp 3-26, 1991.

[Kristiansen, 07] Kristiansen, R., Trætteberg, H.: Model-based user interface design in the context of workflow models. Lecture Notes in Computer Science. 4849:227-239. 2007

[Limbourg, 04a] Limbourg, Q. and Vanderdonckt, J.: Comparing Task Models for User Interface Design. The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates, Mahwah, 2004. pp. 135-154.

[Limbourg, 04b] Limbourg, Q., Vanderdonckt, J., UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence, in Matera, M., Comai, S. (Eds.), «Engineering Advanced Web Applications», Rinton Press, Paramus, 2004, pp. 325-338.

[Lemaigre, 08] Lemaigre, Ch.,Guerrero, J., Vanderdonckt, J., Interface Model Elicitation from Textual Scenarios, Proc. of 1st IFIP TC 13 Human-Computer Interaction Symposium HCIS'2008 (Milan, 8-9 September 2008), P. Forbrig, F. Paterno, A.M. Pejtersen (eds.), International Federation of Information Processing, Vol. 272, Springer, Boston, 2008, pp. 53-66.

[Mandviwalla, 94] Mandviwalla, M., Olfman, L.: What do groups need? A proposed set of generic groupware requirements, ACM Transactions on Computer-Human Interaction, Vol. 1, No. 3, pp. 245 – 268, 1994.

[Montero, 06] Montero, F., López-Jaquero, V.: IdealXML: An Interaction Design Tool-A Task-Based Approach to User Interfaces Design, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006 (Bucharest, 6-8 June 2006), Chapter 20, Springer-Verlag, Berlin, 2006, pp. 245-252.

[OMG, 06] Object Management Group: Business process modeling notation specification, final adopted specification dtc/06-02-02 (2006).

[Paternò, 99] Paternò, F.: Model-based design and evaluation of interactive applications. Applied Computing, Springer. 1999.

[Paternò, 02] Paternò, F., Santoro, C.: One model, many interfaces. In Ch. Kolski and J. Vanderdonckt, Eds. Proceedings of the 4 International Conference on Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 May 2002), pp. 143-154, Dordrecht, 2002. Kluwer Academic Publishers.

[Eisenstein, 00] Eisenstein, J., Vanderdonckt, J., Puerta, A.: Adapting to Mobile Contexts with User-Interface Modeling, Proc. of 3rd IEEE Workshop on Mobile Computing Systems and Applications WMCSA'2000 (Monterey, 7-8 December 2000), IEEE Press, Los Alamitos, 2000, pp. 83-92.

[Russell, 04] Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Data Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven (2004).

[Russell, 05] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M. and Edmond, D.: Workflow Resource Patterns. In the 17th Conference on Advanced Information Systems Engineering (CAISE'05). Porto, Portugal. 13-17, June.

[Stavness, 04] Stavness, N., Schneider, K.: Supporting Workflow in User Interface Description Languages. In Proc. Workshop on Developing User Interface with XML: Advances on User Interface Description Languages, AVI 2004, Gallipoli, Italy, 2004.

[WfMC, 06] Workflow Management Coalition, 2006, http://www.wfmc.org/

[van der Aalst, 98] van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers. Vol. 8, No. 1, February 1998, pp. 21-66.

[van der Aalst, 02] van der Aalst, W.M.P., van Hee, K.: Workflow Management: Models, Methods, and Systems. The MIT Press, Cambridge. 2002

[van der Aalst, 03] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases 14, 3 (2003) 5–51.

[van der Aalst, 05] van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems 30 (2005) 245—275

[van der Veer, 99] van der Veer, G.C. and van Welie, M.: Groupware Task Analysis. Tutorial notes for the CHI99 workshop "Task Analysis Meets Prototyping: Towards Seamless UI Development". 16th May 1999, Pittsburgh PA, USA.

[Vanderdonckt, 05] Vanderdonckt, J., A MDA-Compliant Environment for Developing User Interfaces of Information Systems, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), O. Pastor & J. Falcão e Cunha (eds.), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31.

[W3C, 05] W3C: State Chart XML (SCXML): State Machine Notation for Control Abstraction 1.0. W3C Working Draft 5 July 2005. http://www.w3.org

[Wodtke, 97] Wodtke, D., Weikum, G.: A Formal Foundation for Distributed Workflow Execution Based on State Charts. In Proc. of the 6th International Conference on Database Theory (ICDT '97), Springer-Verlag, LNCS Series, p. 230-246, 1997.